#11

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant	:	Nouri, et al.	Group Art Unit 2758
Appl. No.	:	08/942,333	
Filed	:	October 1, 1997	
For	:	SYSTEM FOR RESETTING A SERVER	
Examiner	:	Philip Tran	

DECLARATION UNDER 37 C.F.R. § 131 TO OVERCOME JOHNSON

1. This declaration is to establish the status of the invention in the above-captioned U.S. patent application in the United States on August 16, 1996, which is the effective date of U.S. Patent No. 5,857,074, entitled "Server Controller Responsive to Various Communication Protocols for Allowing Remote Communication to a Host Computer Connected Thereto", to Johnson, which was cited by the Examiner against the above-captioned application.

2. We are the named joint inventors of the described subject matter and all claims in the above-referenced application.

3. We have read the Office Action mailed January 26, 1999 (Paper No. 7) regarding the above-captioned application.

4. We developed our invention as described and claimed in the subject application in this country, as evidenced by the following events:

a. By at least November 1995, I, Karl Johnson, had conceived of a control diagnostic and monitor subsystem for a server system. A document, entitled "Raptor System: A Bird's Eye View, Version 0.99", was written at least as early as November 2, 1995, as evidenced by the document date. A copy of the cover page, and pages 8 and 9 of document is attached as **Exhibit A**. The control diagnostic and monitor subsystem was

to control various system aspects, including system reset, without the use of physical switches.

b. By at least January 1996, I, Karl Johnson, had conceived of using a network of microcontrollers as the monitoring and control hardware of the subject invention. A document, entitled "Raptor Wire Service Architecture, Version 1.0" ("Wire Architecture"), was written at least as early as January 23, 1996, as evidenced by the document date. A copy of the cover page, pages 2-4, 7-8 and 20-25 of Wire Architecture is attached as **Exhibit B**. Page 2 describes a generic Wire Service message format utilized by the invention. Pages 3-4 describes a bit data type, a byte data type, and a string data type used for data transfer, while pages 7-8 describe an event data type for alerting external interfaces (such as the remote interface) of events in the Wire Service (microcontroller network). Wire Architecture also discloses a Wire Service Network Memory Map of all Wire Service (microcontroller network) addressable entities at pages 20-25 of the document. For example, page 22 of the Wire Architecture indicates several of the addressable entities associated with the server (an example of a first computer) power supplies. These entities include an analog measure of the voltage for the main power supplies and a power supply DC OK status which could be monitored by the Remote Interface microcontroller and/or passed on to the Recovery Manager at the client (an example of a second computer). These and other entities described at pages 22-23 may be utilized by the system reset operation.

c. The server power operations were documented in a specification entitled "Raptor Theory of Power On Operation, Version 1.0" ("Power Operation") , written at least as early as March 1, 1996, as evidenced by the document date. Wei Wu, employed by our employer NetFRAME Inc., prepared this document based upon information provided by me, Karl Johnson. A copy of the cover and pages 1-5 of Power Operation is attached as **Exhibit C**. Page 1 shows a block diagram of the relationship between Wire Service, the System hardware, the BIOS, the Operation system, and the Application. Page 1 also describes that Wire Service resets the system, and discusses BIOS operations, including cold start and warm start.

d. By at least April 1996, I, Karl Johnson had conceived of an architecture for the remote interface module. The remote interface module may be incorporated in or on the

server enclosure. The remote interface communicates with the server microcontroller network and with external computers via a RS-232 port, for example. The remote interface also provides remote power from a remote interface power supply that is independent of the server power supply. A schematic, entitled "Schematic of Raptor Remote Board, Revision 01", was drawn at least as early as April 1, 1996, as evidenced by the document date. A copy of sheets 1 of 2 and 2 of 2 of the schematic is attached as **Exhibit D**. A remote interface microcontroller (PIC16C65), memory and RS-232 interface are shown on sheet 1. The top left portion of sheet 2 generates the independent power. The independent remote interface power supply is not shown but connects to connector J2 on pins 1 and 2. A RJ45 connector P1 (also on sheet 2) provides an interconnection point between the remote interface microcontroller and the server microcontroller network.

e. The architecture for the remote interface was documented in a specification entitled "Remote Interface Board Specification, Revision 2" ("RIB Specification"), written at least as early as June 21, 1996, as evidenced by the document date. A copy of the RIB Specification is attached as **Exhibit E**. The RIB Specification recites at page 3 that the RIB is an interface between Raptor Wire Services (the microcontroller network) and an external modem. The system status and commands are passed through the RS232 connection at the modem side to the Wire Services bus controlled through the on-board microcontroller.

f. By at least October 1996, we developed a revised version of the architecture for the network of microcontrollers. A document, entitled "Raptor Wire Service Architecture, Version 1.3" ("Wire Architecture"), was written at least as early as October 3, 1996, as evidenced by the document date. A copy of the cover and pages 1, 10 and 36-37 of Wire Architecture is attached as **Exhibit F**. Page 1 has a Wire Services hardware block diagram which shows how the remote interface connects with the microcontroller network so as to obtain system status and control various entities. Page 10 describes the event data type for alerting external interfaces of events in the microcontroller network. The event data type were revised since the earlier version of the document. Pages 36-37 describe the Wire Service Remote Serial protocol used to communicate microcontroller network messages across a serial link from the remote interface microcontroller attached to the server to a Wire Service remote management processor at the second computer.

g. A schematic, entitled "Schematic of P6 Mother Board, Revision 54", was written at least as early as October 24, 1996, as evidenced by the document date. A copy of sheet 42 of 60 for the Speed Fan Controller section is attached as **Exhibit G**. This section includes the server end of the RJ45 connector (labeled P13) which interconnects the microcontroller network with the remote interface. The RJ45 connector also receives the independent power at pin 5 from the remote interface. Pin 5 of the P13 connector then feeds a diode D15, the output of which is the Bias_5V (independent) power for the server.

h. By at least May 1997, we developed a revised version of the architecture for the remote interface. A schematic, entitled "Schematic of Raptor Remote Board, Revision 50", was drawn at least as early as May 6, 1997. A copy of sheets 1 of 2 and 2 of 2 is attached as **Exhibit H**.

i. By at least May 13, 1997, which is the filing date of a provisional patent application for the subject matter, I, Ahmad Nouri had written a document entitled "Maestro Recovery Manager Analysis - Problem Statement". This document was included as a portion of the provisional patent application filed on May 13, 1997. A copy of pages 1-3 of the document is attached as **Exhibit I**. The Maestro Recovery Manager provides a user-friendly graphical user interface for controlling and receiving information from the microcontrollers and the microcontroller network. The reset operation is initiated by use of the Maestro Recovery Manager as described on page 3 of the document.

5. I, Karl S. Johnson, am listed as an inventor on a provisional Patent Application No. 60/046,397, filed May 13, 1997, which is a priority application for the subject application. I, Ahmad Nouri, am listed as the inventor on a provisional Patent Application No. 60/046,326, filed May 13, 1997, which is a priority application for the subject application.

6. We are the listed inventors on the subject regular patent applications filed on October 1, 1997.

7. All acts leading to the reduction of practice were performed in the United States.

8. This declaration is submitted prior to a final rejection.

Penalty of Perjury Statement

We declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful, false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful, false statements may jeopardize the validity of the application or any patent resulting therefrom.

Dated:_____ By:_____
Ahmad Nouri

Dated: April 24, 1999 By: Karl S Johnson
Karl S. Johnson

RJS-2090.DOC
042299

Raptor System

A Bird's Eye View

Karl Johnson (KJ)

Revision 0.99

November 2, 1995

Control Diagnostic and Monitor Subsystem

The control of Raptor is completly "Fly By Wire" - i.e. no physical switch directly controls any function and no indicator is directly controlled by system hardware. All such functions, refered to as Out Of Band functions[1], are controlled through a Control Diagnostic and Monitor (CDM) subsystem implemented by small distributed CDM processors connected on a 400 kbs I^2C serial bus (the CDM Bus). Critical CDM components are powered by the bias power from any of the power supplies, which means that CDM basic CDM functions are avaiable so long as A/C power is available at the input to any of the power supplies. The CDM subsystem supervises or monitors the following system features:

[1]See Glossary

- Power supplies - Presence, status, A/C good, Power On/Off and Output voltages.
- Environment - Ambient and exhaust temperatures, Fan speed, speed control, Fan fault and Overtemp indicators.
- Processor - CPU Presence, Power OK, Overtemp and Fault, NMI control, System reset, Memory type/location and Bus/Core speed ratio.
- I/O - I/O Canister insertion/removal and status indicator, PCI card presence, PCI card power and Smart I/O processor Out Of Band control.
- Historical - Log of all system events, Character mode screen image, and Serial Numbers.

Control of CDM functions is either intrinsic (i.e. CDM components act automatically to perform the function, such as when a fan fails, the remaining fan has its speed increased and the fan failure indicator is lit) or external (i.e. the CDM subsystem gets external input requesting the function). External functions can be initiated from either the system interface port by one of the P6 processors or through the RS232 serial CDM interface connected to the CDM External port.

The CDM subsystem remote access feature provides for remote management of the system when the Operating System is not available[2].

[2]See section on Remote Management.

Raptor Wire Service Architecture

Version 1.0

1/23/96

Prepared for
Raptor Implementation Group

by
Karl Johnson (KJ)

Generic Wire Service I²C Message Format

The generic Wire Service message format is designed for easy encode/decode and reliability. It is not necessarily always the most compact representation possible

Master Asserts START

Offset	MSB	LSB	
Byte 0	Slave Address	0	0 means I²C write to slave
Byte 1	Data Type	R/W	R/W = 0 Mem Write / 1 Mem Read
Byte 2	Sub-Address		Most Significant Byte of Address
Byte 3	Sub-Address (Continued)		Least Significant Byte of Address
Byte 4	Length of Data		Length write or read buff size
Byte 5	Data		Present only for memory write
:	:		:
Byte N	Checksum		

Master repeats START

Byte 0	Slave Address (same)	1	1 means I²C read from slave
Byte 1	Length of Data		Length of Data only (N-3)
Byte 2	Data		
:	:		
Byte N-1	Status		Status 0=Success otherwise Failure
Byte N	Checksum		

Data Type Descriptions

Each data type description includes a rational for its existence and an example simple protocol message.

Bit Type

The bit data type is to be used for simple logic valued items (TRUE/FALSE, ON/OFF, etc.)

Read Bit Message

Request

Slave Address	Type/RW	Bit Addr MSB	Bit Addr LSB	Request 1	Check Byte

Response

Slave Address	Length 1	Bit Value (0/1)	Status 0/Success	Check Byte

Write Bit Message

Request

Slave Address	Type/RW	Bit Addr MSB	Bit Addr LSB	Length 1	Bit Value (0/1)	Check Byte

Response

Slave Address	Length 0	Status 0/Success	Check Byte

Byte Type

The byte data type is to be used for single byte valued items (0-255)

Read Byte Message

Request

Slave Address	Type/RW	Byte Addr MSB	Byte Addr LSB	Request 1	Check Byte

Response

Slave Address	Length 1	Byte Value (0-255)	Status 0/Success	Check Byte

Write Byte Message
Request

Slave Address	Type/RW	Byte Addr MSB	Byte Addr LSB	Length 1	Byte Value (0-255)	Check Byte

Response

Slave Address	Length 0	Status 0/Success	Check Byte

String Type

The String type is designed to handle data that is best organized as variable length strings of 0 to 255 bytes. Internal allocation of string storage for each string may be less than 255 bytes, and writing a string longer than available storage will return an error.

Read String Message
Request

Slave Address	Type/RW	String Addr MSB	String Addr LSB	Request 0-255	Check Byte

Response

Slave Address	Length N	String Data 1	. . .	String Data N	Status 0/Success	Check Byte

Write String Message
Request

Slave Address	Type/RW	String Addr MSB	String Addr LSB	Length N	String Data 0	. . .	String Data N
Check Byte							

Response

Slave Address	Length 0	Status 0/Success	Check Byte

The addressing of log entries has some special characteristics.
1) Reading address 65565 (0xffff) is special - It represents the address of the latest entry in the log.
2) Reading address 65564 (0xfffe) is also special - It represents the address of the earliest available entry.

3) The address of real log entries wraps at 65519 (0xffef). The next sequential entry after 65519 is 0.
4) The address of is ignored on write and the next available entry is written.
5) To read the entire log in forward time order, read entry at address 65564. This returns the first log entry along with its actual log address. Increment that address by one and read that entry. Repeat the last step until status indicates failure.
6) To read the entire log in reverse time order, read entry at address 65565. This returns the last log entry along with its actual log address. Decrement that address by one and read that entry. Repeat the last step until status indicates failure.
7)To keep a complete external copy of the log, first read the entire log in forward time order and remember the last valid entry. Then periodically read forward from the remembered last valid entry to the end and add that to the external copy.

Event Type

The event data type is to be used for alerting external interfaces of events in the Wire Service network. Event memory is organized as a queue. The queue will probably be quite small (< 20 Events). Writing an event places the event ID at the next available entry, unless the last queue entry would be written by this event. In that case, the last queue entry is a Queue Overflow Event and the write fails. This allows the external interface to realize that events were lost and it should scan for any changes in data.
Reading the event type returns requested number of events in the queue or the entire queue which ever is less and removes them from the queue.

Read Event Message
Request

Slave Address	Type/RW	N/A	N/A	Request 1-255	Check Byte

Response

Slave Address	Length N	Event ID 1	. . .	Event ID N	Status 0/Success	Check Byte

Write Event Message
Request

Slave Address	Type/RW	N/A	N/A	Leagth 1	Event ID	Check Byte

Response

Slave Address	Length 0	Status 0/Success	Check Byte

Possible Event Types:
CPU Status Change
Power Status Change
Canister Status Change
Fan Status Change

Screen Type

The screen data type is to be used for communication of character mode screen information from the system BIOS to remote management interface.

Read Screen Message
Request

Slave Address	Type/RW	S Addr MSB	S Addr LSB	Request 1-255	Check Byte

Response

Slave Address	Length N	Screen Data 1	. . .	Screen Data N	Status 0/Success	Check Byte

Write Screen Message
Request

Slave Address	Type/RW	S Addr MSB	S Addr LSB	Length N	Screen Data 1	. . .	Screen Data n
Check Byte							

Response

Slave Address	Length 0	Status 0/Success	Check Byte

Wire Service Network Memory Map

This section defines the Wire Service Network Memory Map for the first Raptor system. Its purpose is to identify all Wire Service addressable entities and describe their function and any special information about them.

This section is incomplete yet and only a small incomplete sample is supplied. (Although some of the more complicated ones are described.)

The address format is "pp:aaaa", where "p" is the processor ID (hexadecimal) of the Wire Service Processor where the data resides and "aaaa" is the hexadecimal address or address range for the data.

Name	Type	Address	Description	Notes
WS_DESC_Pn	STRING	0n:0000	Wire Service Processor Type/Description	
WS_REV_Pn	STRING	On0001	Wire Service Software Revision/Date Info	
WS_SB_FAN_HI	BIT	03	System Board Fans HI	Controls S1_FAN_HI. Set on 0->1 transition of WS_SB_FAN_LED. Cleared by other software
WS_SB_FAN_LED	BIT	03	System Board Fan Fault LED	Controls S1_SBFAN_LED. It is set whenever any WS_SB_FANFAULTn is set. Log 0->1 transition
WS_SB_BUSCORE	BYTE	03	System Board BUS/CORE speed ratio to use on reset	Value is asserted on S1_BC_DS[0-3] unless reading DIMM types. Set to 0 on power on.
WS_SYS_LCD	STRING	03	Value to display on LCD	For a Nx2 display the first N bytes display on top line and the second N bytes display on the bottom line. Manipulates S1_LCD_D[0-7], S1_LCD_RS, S1_LCD_ENA, S1_LCD_RW
WS_SB_FAN1	BYTE	03	System Board Fan 1 speed	Approximately every second a fan is selected by S1_FAN_SEL[0-2] and monitored via S1_FAN_TP driving a counter for a known period of time. The counter is then loaded into the appropriate fan speed. If WS_SB_FAN_HI is not set then the speed is compared against WS_SB_FAN_LOLIM. If fan is slow set appropriate WS_SB_FANFAULTn otherwise clear it
WS_SB_FAN2	BYTE	03	System Board Fan 2 speed	
WS_SB_FAN3	BYTE	03	System Board Fan 3 speed	
WS_SB_FAN4	BYTE	03	System Board Fan 4 speed	
WS_SB_FANFAULT1	BIT	03	System Board Fan 1 Faulted	
WS_SB_FANFAULT2	BIT	03	System Board Fan 2 Faulted	

WS_SB_FANFAULT3	BIT	03	System Board Fan 3 Faulted	
WS_SB_FANFAULT4	BIT	03	System Board Fan 4 Faulted	
WS_SB_FAN_LOLIM	BYTE	03	Fan speed low speed fault limit	Set to ??? on power on
WS_SB_DIMM_SEL	BYTE	03	The DIMM select bits to use when reading DIMM_TYPE	The low order 4 bits are the select bits to use when WS_SB_DIMM_TYPE is read.
WS_SB_DIMM_TYPE	BYTE	03	The type of DIMM in the DIMM_SEL position	When read asserts value of WS_SB_DIMM_SEL on S1_BC_DS[0-3] and then returns value of S1_DIMM_D[0-7]
WS_SB_FLASH_ENA	BIT	04	Indicates FLASH ROW write enabled	Set/Cleared by debounced 0->1 transition of S2FLASH_SW. Controls state of S2_FLASH_WE and S2_FLASH_LED.
WS_SB_FRU_FAULT	BIT	04	Indicates the FRU status	At power on starts at 1. Controls S2_SBFLT_LED[0-1] for bicolor LED colors 0=Green 1=Amber, Cleared by other software
WS_SYS_OVERTEMP	BIT	04	Indicates Overtemp fault	At power on is set. Controls S2_OVRTMP_LED. Controlled by wire service backplane processor.
WS_SB_JTAG	BIT	04	Enables JTAG chain on system board	Clear at power on. Controls S2_SB_JTAG
WS_SB_CPU_PRES	BYTE	04	CPU Presence bits (LSB = CPU1)	Assemble from S2_PRES_CPU[1-4]
WS_SB_CPU_ERR	BYTE	04	CPU Error bits (LSB = CPU1)	Assemble from S2_ERROR_CPU[1-4]
WS_SB_CPU_TEMP	BYTE	04	CPU Thermal fault bits (LSB = CPU1)	Assemble from S2_TEMP_CPU[1-4]
WS_SB_CPU_POK	BYTE	04	CPU Power OK (LSB = CPU1)	Assemble from S2_POK_CPU[1-4]
WS_NMI_MASK	BYTE	04	CPU NMI processor mask (LSB=CPU1)	Defaults to all ones on power up
WS_NMI_REQ	BIT	04	NMI Request bit	When set pulse S2_NMI_CPUn corresponding to each bit set in WS_NMI_MASK. Then clear request bit. Log Action
WS_SYSFAULT	BIT	04	System Fault Summary	This bit is set if any faults detected in the system. Controls S2_SYSFLT_LED Bits scanned WS_SP_CPU_FAULT, WS_SB_FRU_FAULT (other faults?)
WS_SB_CPU_FAULT	BIT	04	CPU Fault Summary	This bit is set if ((WS_SB_CPU_ERR | WS_SB_CPU_TEMP | ~WS_SB_CPU_POK) & ~WS_SB_CPU_PRES) != 0. Log 0->1 transition with CPU bytes.

WS_BP_P5V	BYTE	02	Analog Measure of +5 volt main supply	read from S4_VOLTS_P5v
WS_BP_P3V	BYTE	02	Analog Measure of +3.3 volt main supply	read from S4_VOLTS_P3v
WS_BP_P12V	BYTE	02	Analog Measure of +12 volt main supply	read from S4_VOLTS_P12v
WS_BP_P5V	BYTE	02	Analog Measure of -12 volt main supply	read from S4_VOLTS_N12V
WS_SYS_CAN_PRES	BYTE	02	Presence bits for canisters (LSB=1, MSB=8)	controlled by S4_PSN_CAN[1-8]. A previous value byte needs to be maintained so canister transitions can be recognized. Previous value initialized to zero. Periodic monitor scans for new canisters. When new canister is recognized read full serial data and store in WS_SYS_CAN_SERIALn then log and send event
WS_SYS_PS_PRES	BYTE	02	Presence bits for power supplies (LSB=1, MSB=3)	controlled by S4_PSN_PS[1-3]. A previous value byte needs to be maintained so power supply transitions can be recognized. Previous value initialized to zero. Periodic monitor scans for new power supplies. When new power supply is recognized read full serial data and store in WS_SYS_PS_SERIALn then log and send event
WS_SYS_PS_ACOK	BYTE	02	Power supply ACOK status (LSB=1, MSB=3)	controlled by S4_ACOK_PS[1-3]. A previous value byte needs to be maintained so power supply transitions can be recognized. Previous value initialized to zero. Periodic monitor scans for changes is ACOK and sends events
WS_SYS_PS_DCOK	BYTE	02	Power supply DCOK status (LSB=1, MSB=3)	controlled by S4_DCOK_PS[1-3]. A previous value byte needs to be maintained so power supply transitions can be recognized. Previous value initialized to zero. Periodic monitor scans for changes is ACOK and sends events
WS_SYS_BP_TYPE	BYTE	02	Type of system backplane currently only two types Type 0 = 4 canister (small) and Type 1= 8 canister (large)	controlled by S4_BP_TYPE
WS_SYS_TEMP_SB1	BYTE	02	Temperature of system board position 1	controlled by reading Dallas temperature transducers connected to serial bus on S4_TEMP_SDA and S4_TEMP_SCL
WS_SYS_TEMP_SB2	BYTE	02	Temperature of system board position 2	
WS_SYS_TEMP_BP1	BYTE	02	Temperature of backplane position 1	
WS_SYS_TEMP_BP2	BYTE	02	Temperature of backplane position 2	

WS_SYS_TEMP_WARN	BYTE	02	Warning temperature. Initialized to ???	If any WS_SYS_TEMP_xxn exceeds this value, log, send event, set WS_SYS_OVERTEMP
WS_SYS_TEMP_SHUT	BYTE	02	Shutdown temperature. Initialized to ???	If any WS_SYS_TEMP_xxn exceeds this value, log and clear WS_SYS_POWER
WS_SYS_REQ_POWER	BIT	02	Set to request main power on	
WS_SYS_POWER	BIT	02	Controls system master power S4_POWER_ON	When this bit is set 0->1 set S4_POWER_ON, WS_SYS_RUN = 0, WS_SYS_RSTIMER = 5 and log. When this bit is cleared clear S4_POWER_ON and log.
WS_SYS_RSTIMER	BYTE	02	Used to delay reset/run until power stabilized	Counts down to 0 at 10 counts per second. When 1->0 transition sets WS_SYS_RUN.
WS_SYS_RUN	BIT	02	Controls the system halt/run line S1_OK_TO_RUN.	If this bit is cleared, clear S1_OK_TO_RUN and log. If this bit is set, set S1_OK_TO_RUN and log.
WS_CAN_POWER	BIT	2x	Controls canister PCI slot power	When set then set S5_P5V_ENA[1..4],S5_P12V_ENA in that order with small (about 1 ms) delay between each, then log. When cleared then clear S5_P12V_ENA, S5_P5V_ENA[1..4] then log
WS_CAN_PCI_PRESENT	BYTE	2x	Reflects PCI card slot[1..4] presence indicator pins (MSB to LSB) 4B,4A,3B,3A,2B,2A,1B, 1A	Reflects data from S5_PRSNT_S[1..4][A/B]
WS_CAN_S5_PRESENT	BIT	2x	Indicates the presence of something in slot 5	Reflects S5_PRSNT_S5
WS_CAN_S5_SMART	BIT	2x	Indicates something other than a passive board in slot 5	On power up attempt to read Dallas serial number chip using S5_PSN_S5. If present set this bit and read full serial data and store in WS_SYS_CAN_IOP_SERIALn
WS_CAN_FAN_HI	BIT	2x	Canister Fans HI	Controls S1_FAN_HI. Set on 0->1 transition of WS_SB_FAN_LED. Cleared by other software
WS_CAN_FAN_LED	BIT	2x	Canister Fan Fault LED	Controls S5_CANFAN_LED. It is set whenever any WS_CAN_FANFAULTn is set. Log 0->1 transition
WS_CAN_FANFAULT1	BIT	03	Canister Fan 1 Faulted	
WS_CAN_FANFAULT2	BIT	03	Canister Fan 2 Faulted	

WS_CAN_FAN1	BYTE	2x	Canister Fan 1 speed	Approximately every second a fan is selected by S5_FAN_SEL0 and monitored via S5_FAN_TP driving a counter for a known period of time. The counter is then loaded into the appropriate fan speed. If WS_CAN_FAN_HI is not set then the speed is compared against WS_CAN_FAN_LOLIM. If fan is slow set appropriate WS_CAN_FANFAULTn otherwise clear it
WS_CAN_FAN2	BYTE	2x	Canister Fan 2 speed	
WS_CAN_FAN_LOLIM	BYTE	2x	Fan low speed fault limit	Set to equivalent of xxx RPM on power on
WS_CAN_JTAG_ENA	BIT	2x	Enable JTAG TMS chain for canister	Copy set value to S5_CAN_JTAG
WS_CAN_NMI_S5	BIT	2x	NMI card in slot 5	when set, pulse S2_NMI_S5
WS_RI_CD	BIT	11	Status of Remote Port Modem CD	Follows S6_MODEM_CD
WS_RI_DTR	BIT	11	State of Remote Port Modem DTR	Controls S6_MODEM_DTR
WS_RI_DSR	BIT	11	Status of Remote Port Modem DSR	Follows S6_MODEM_DSR
WS_RI_RTS	BIT	11	Status of Remote Port Modem RTS	Controls S6_MODEM_RTS
WS_RI_CTS	BIT	11	Status of Remote Port Modem CTS	Follows S6_MODEM_CTS
WS_RI_CALLOUT	BYTE	11	Controls Call out Script activation	If written to it initiates Call out sequence programmed in WS_SYS_CALL_SCRIPT passing value as argument to script. Log it (Format of Script Programs TBD)
WS_RI_EVENTS	EVENT	11	Remote Interface Event Queue	See Event Data type description in prior section.
WS_SI_EVENTS	EVENT	10	System Interface Event Queue	See Event Data type description in prior section.
WS_SYS_LOG	LOG	01	System Log	The system log kept in NVRAM (See LOG data type in previous section)
WS_SYS_SCREEN	SCREEN	01	System Screen	A copy of the most recent character mode screen from the system video display (See SCREEN data type in previous section)
WS_SYS_SB_SERIAL	STRING	01	Last known System Board serial data	
WS_SYS_BP_SERIAL	STRING	01	Last known Back Plane serial data	
WS_SYS_RI_SERIAL	STRING	01	Last known Remote Interface serial data	

WS_SYS_CAN_SERIAL[1-8]	STRING	01	Last known Canister [1-8] Serial data	May be zero length if no canister ever seen
WS_SYS_IOP_SERIAL[1-8]	STRING	01	Last known IOP in Canister [1-8] Serial data	May be zero length if no canister ever seen or current canister has no IOP
WS_SI_QUEUE	QUEUE	01	Queue of data going to System Interface	See Queue data type in previous section
WS_RI_QUEUE	QUEUE	01	Queue of data going to Remote Interface	See Queue data type in previous section
WS_SYS_XDATA	BYTE ARRAY	01	Byte Array for storage of arbitrary external data in NVRAM	Wire Service just maintains this data area and is unaware of the meaning of any data stored in it.
WS_SYS_EXT_KB	BYTE	01	Size of the WS_SYS_XDATA in kilobytes	Necessary for memory management of the data area

NetFRAME®

RAPTOR THEORY OF POWER ON OPERATION

Version 1.0

March 1, 1996

Prepared by Wei Wu

1. Introduction:

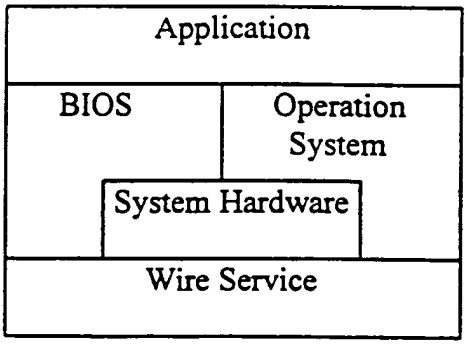
The purpose of this document is to describe the power on operation sequences of the RAPTOR system BIOS and Wire Service. The operation sequences includes that the system at the no power, after the power on, the cold start, the warm start, the peripheral initialization, the system setup, and at the power down. Two other related subjects, BIOS events code and local LCD display, are also included in this document.

The RAPTOR Wire Service is acted as a system administrator. It monitors the signals which come from control switches, temperature sensors, and remote terminals. By the signals, the Wire Service turns on/off the power to the mother board and back plane, resets the system, turns the system cooling fans to high, low, or off, provide system operating parameters to BIOS, take the POST events information from BIOS, and sends those data to the display panel and remote terminals.

The BIOS will be first called when the mother board is powered up or after reset. The function of the BIOS is to detect system hardware configuration, test and initialize the hardware, send POST events to the Wire Service, determining the operation system boot up driver, and provide basic input and output routines to the operation system.

After BIOS finished its POST task, the operation system start to boot. It checks the hardware configuration data from the BIOS, initializes hardware if it is necessary, installs device drivers, send events to the Wire Service, and response the requests for the Wire Service.

The following block diagram shows the relationship among the system hardware, BIOS, Wire Service, operation system, and application software.



2. System No Power

When the system power off, there is no power to the RAPTOR mother board, back plane, and canisters. But, the Wire Service is still powered as system administrator,. The Wire Service monitors power on/off switch or the power on signal from the remote terminal.

3. System Power On

As the Wire Service detects that the power on/off switch is on or the turn on signal from the remote terminal(WS_SYS_POWER is set), it log the power on requested and temperature data. If the WS_SYS_OVERTEMP is set, send over temperature message to the remote terminal and stop. Otherwise, it set S4_POWER_ON, set WS_SYS_RSTIMER to 5, and turn on the power and cooling fan on the mother board, back plane, and canisters.

When the mother board is powered up, the BIOS POST routine is called. The BIOS first initialize the PCI-ISA bridge and Wire Service driver. Then, it check cold/warm start flag. If the system is reset by keyboard, the Warmstart procedure is called. Otherwise, the Coldstart procedure is called.

The Wire Service will monitor the hardware temperature, switches on the switch control panel, signals from the remote terminal. It also log the BIOS POST events.

The figure 3.1 on next page is the flow chart of the POST procedure.

4. BIOS POST Coldstart

In the Coldstart POST path, about sixty one (61) BIOS subroutines are called. The procedure of the Coldstart is grouped into following steps:

4.1 CUP Initialization - The POST verify real mode. If the CPU is in protected mode, turn on the address line A20 and pulse the reset line to reset CPU. If CPU is in real mode, get the CPU type and save it in NVRAM.

4.2 DMA\Timer Reset - The POST reset the DMA controllers, disable the video, clear any pending interrupts from the real-time clock, initialize timers, and set segment-register addressibility to 4GB.

4.3 BIO Image Check - The POST verify the ROM BIOS checksums, test 512K base address lines, RAM, and DRAM refresh,

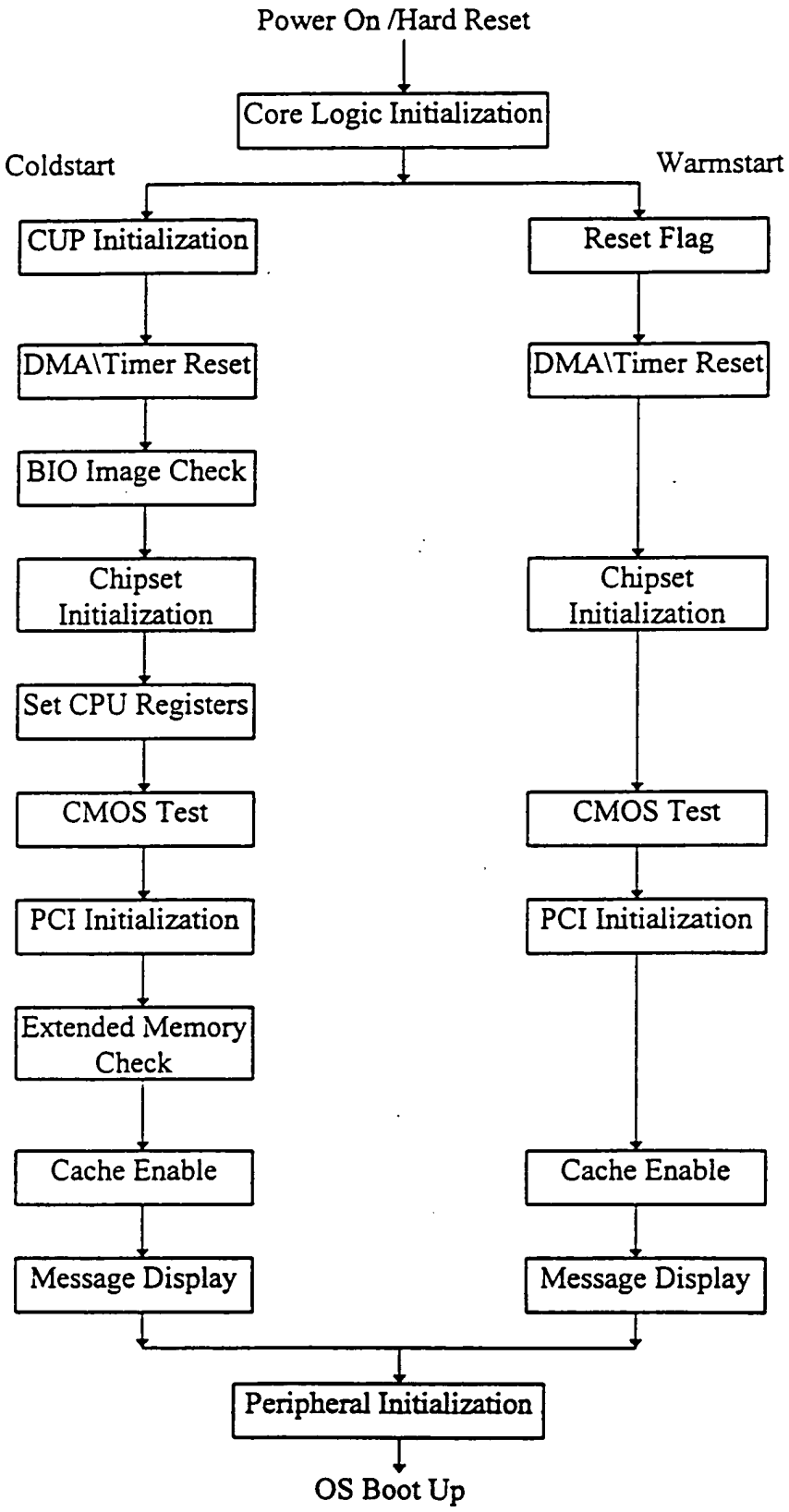


Figure 3.1 the flow chart of the POST procedure

4.4 Chipset Initialization - The POST initialize chip set registers, alternate registers, and caches registers to their initial POST values, initialize external cache, autosize DRAM and cache, clear 512K base RAM, shadow system BIO ROM, and set in-POST flag in NVRAM that indicates the BIOS is in POST.

4.5 Set CPU Registers - The POST initialize CPU registers, enable the CPU cache, initialize keyboard controller, and compute CPU speed.

4.6 CMOS Test - The POST test CMOS RAM, initialize interrupt vectors 0 through 77h to the BIOS general interrupt handler and DMA controller.

4.7 PCI Initialization - The POST initialize manager for PCI option ROMs, initialize PCI bus and devices, check video configuration against CMOS, initialize all video adapters on system, shadow video BIOS ROM, test the keyboard, and test for unexpected interrupts.

4.8 Extended Memory Check - In this step The POST will size and test system and extended memory. The POST first disable CPU cache. It, then, test RAM between 512K and 640K, determine and test the amount of extended memory available, and perform an address lines test on A0 to the amount of memory available.

4.9 Cache Enable - The POST set cache registers to their CMOS values and enable external cache and CPU cache.

4.10 Message Display - This is the last step of the cold start path. The POST display external cache size, shadow messages, non-disposable segments, and error messages if there is error. After that, The POST test the real-time clock, initialize hardware interrupt vectors, and reset segment-register addressibility from 4 GB to normal 64K. Then, the Coldstart path go to the peripheral initialization of the POST.

5. BIOS POST Warmstart

For the Warmstart, the system information are already in the NVRAM. It will skip some POST subroutines such as verify real mode and get CPU type. In the Warmstart path, about forty one (41) subroutine are called. they are grouped to following steps:

5.1 Reset Flag - The POST first clear in-POST flag in NVRAM.

5.2 DMA\Timer Reset - The POST, then, reset the DMA controllers, disable the video, clear any pending interrupts from the real-time clock, initialize timers, and set segment-register addressibility to 4GB.

5.3 Chipset Initialization - The POST initialize chip set registers, alternate registers, and caches registers to their initial POST values, initialize external cache, autosize

DRAM and cache, clear 512K base RAM, shadow system BIO ROM, *enable CPU cache*, and set in-POST flag in NVRAM that indicates the BIOS is in POST.

5.4 CMOS Test - The POST initialize DMA controller, test CMOS RAM, and initialize interrupt vectors 0 through 77h to the BIOS general interrupt handler.

5.5 PCI Initialization - The POST initialize manager for PCI option ROMs, initialize PCI bus and devices, check video configuration against CMOS, initialize all video adapters on system, shadow video BIOS ROM, reset and test the keyboard, and test for unexpected interrupts.

5.6 Cache Enable - The POST set cache registers to their CMOS values and enable external cache and CPU cache.

5.7 Message Display - At the last step of the warm start path, The POST display error messages if there is error, initialize hardware interrupt vectors, and reset segment-register addressibility from 4 GB to normal 64K. Then, the Coldstart path go to the peripheral initialization of the POST.

6. Peripheral Initialization

The conclusion of the POST is the point that Coldstart path and Warmstart path are merged together. The POST will initialize serial/parallel ports, hard/floppy disk drivers, and prepare boot from operation system (OS).

6.1 Port Initialization - The POST first disable address line A20 and onboard COM and LPT ports before testing for presence of external I/O devices. It, then, configure all PnP ISA devices, test and identify RS232 and parallel ports, and re-initialize onboard I/O ports.

6.2 Controllers Initialization - The POST initialize time-outs, key buffer, soft reset flag, floppy and hard disk controllers, mouse, and extended BIOS data area. Then, it setup interrupt vector and present bit in equipment byte, install CD ROM for boot if it is present, and build the MPTABLE for multiprocessor.

6.3 OS Boot Preparation - Before the system boot form OS, The POST enable hardware interrupts, verify that the system clock is interrupting, check the key lock, get system operating parameters from the Wire Service, call SETUP subroutine, store system operating parameters to the Wire Service, clear in-POST flag, clear the screen if BCP option is enable, check the total number of fast disks (ATA and SCSI), and update the bdaFdiskCount.

Now the BIOS is ready to call INT19 to boot from operation system.

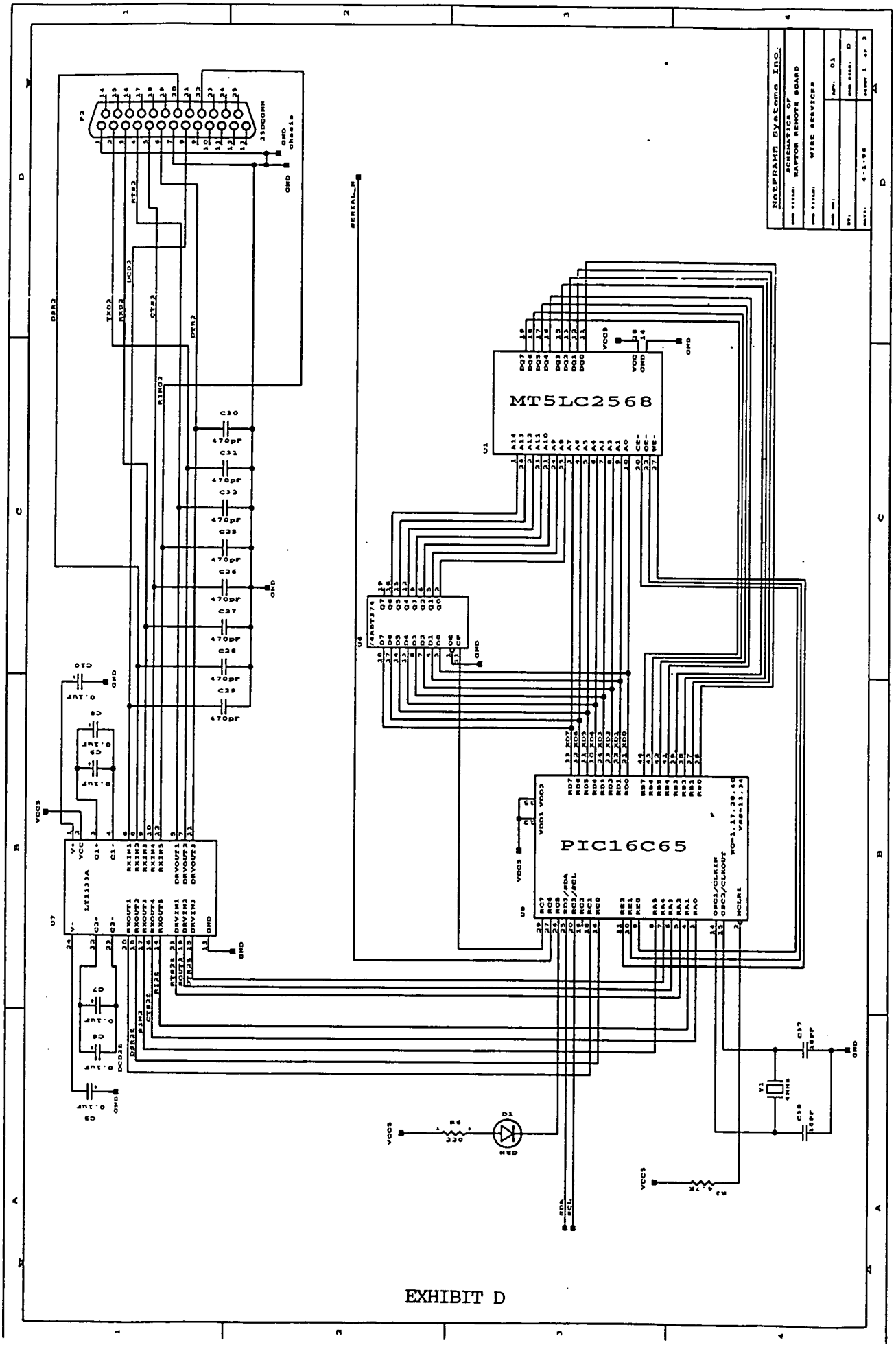
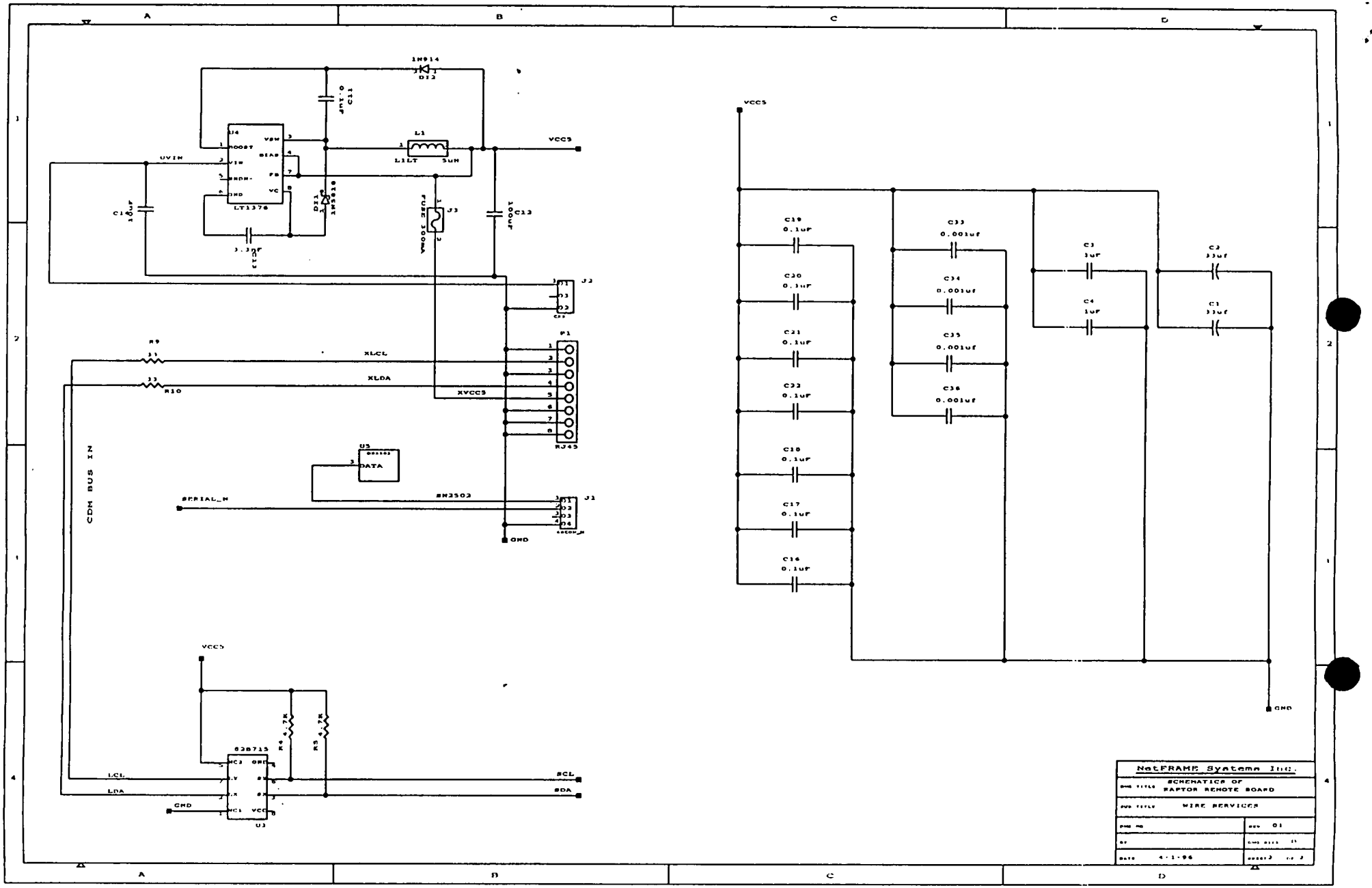

NetFRAME Systems Inc.
SCHEMATICS OF
RAPTOR REMOTE BOARD
WIRE SERVICES
4-3-96
MT5LC2568
PIC16C65
74ST374
LT1133A
25DCONN
SERIAL_N
VCC5
GND

EXHIBIT D


1N914
D12
C11
0.1uF
U4
LT1376
VSW
BOOST
BIAS
VIN
FB
SHDN
VC
GND
UVIN
L1
L1LT
5uH
VCC5
C10
10uF
D11
1N5818
J3
FUSE 300MA
C12
100uF
3.3nF
C13
J2
P1
RJ45
R9
33
R10
33
XLCL
XLDA
XVCC5
CDM BUS IN
U5
DATA
SERIAL_N
2N3502
J1
GND
C19
0.1uF
C20
0.1uF
C21
0.1uF
C22
0.1uF
C18
0.1uF
C17
0.1uF
C16
0.1uF
C33
0.001uF
C34
0.001uF
C35
0.001uF
C36
0.001uF
C3
1uF
C4
1uF
C2
33uF
C1
33uF
VCC5
R4 4.7K
R3 4.7K
83C715
U3
LCL
LDA
GND
SCL
SDA
NetFRAME Systems Inc.
SCHEMATICS OF
RAPTOR REMOTE BOARD
WIRE SERVICES
4-1-96

Remote

Interface Board

Specification

Revision 2
13-000072-01
June 21, 1996

Tahir Sheikh
NetFRAME Systems, Inc.

EXHIBIT E

Index

Overview:

This board is an interface between Raptor Wire Services and an external modem. The system status and commands are passed through the RS232 connection at the modem side to the Wire Services bus, the I2C bus, controlled through an on board PIC16C65. The I2C signals are translated by the PIC16C65 into an eight signal RS232 protocol and passed through a voltage level translator LT1133A, with baud capable of reaching the speed of 120k. A 25 pin D-Sub connector resides on the other side of the voltage level translator.

The system status storage is through a 32Kx8 SRAM, with an external lath for latching the higher addressing bits of the data RAM. A signal powered EPROM is used for storing board ID information.

The board is powered through 7.5V and 700mA supply unit, and is an alternative source for the bias powered partition of the Wire Services. The bias powered block includes an NV-RAM and a PIC16C65 which are resident on the Raptor back plane. The power source is regulated through a high frequency switching regulator.

1.0 Features

The designed features are as follows:

1.1 I2C Interface

The two wires interface is brought from the Raptor and passed to the PIC16C65 using an RJ45. A bus extender 82B715 is connected between the external interface to the local I2C bus. Port C bit 3 is the clocking bit, and Port C bit 4 is the data line.

1.2 RS232 Protocol

The communication with the modem is based on the RS232. Microcontroller PIC16C65 is used to generate the receive and the transmit signals, where the signal levels are transposed to the RS232 levels by the LT1133A. The 3 transmit signals, RTS, SOUT and DTR are from Port A bits 2, 3 and 4, where as the 5 receive signals are from two ports, DCD, DSR from Port C 1,0 and SIN, CTS and RI from Port A 5, 0, 1.

The 25 pin RS232 pin connection is used instead a 9 pin connector, since this type of connector is more common than the other. All the extra pins are no connect except the pins 1 and 7, where pin 1 is chassis ground and pin 7 is a signal ground.

The connection through LT1133A can be run up to 120k Baud and is ESD protected to +/- 10kV.

The short voltage at the output can be +/- 30V and is isolated to the forward direction only.

1.3 PIC16C65 and 32Kx8

A 32Kx8 SRAM is available for storage and transfer between the internal Wire Services and the external remote interface. Port D is the address port, while an external 74ABT374 is for expanding the address range to 15 bits. Port B is the data bus for the bi-directional data interconnect. Port E is for the SRAM enable, output tristate and the write control signals.

The PIC16C65 is designed for a frequency of 12MHz. An LED is also connected to the Port C bit 5.

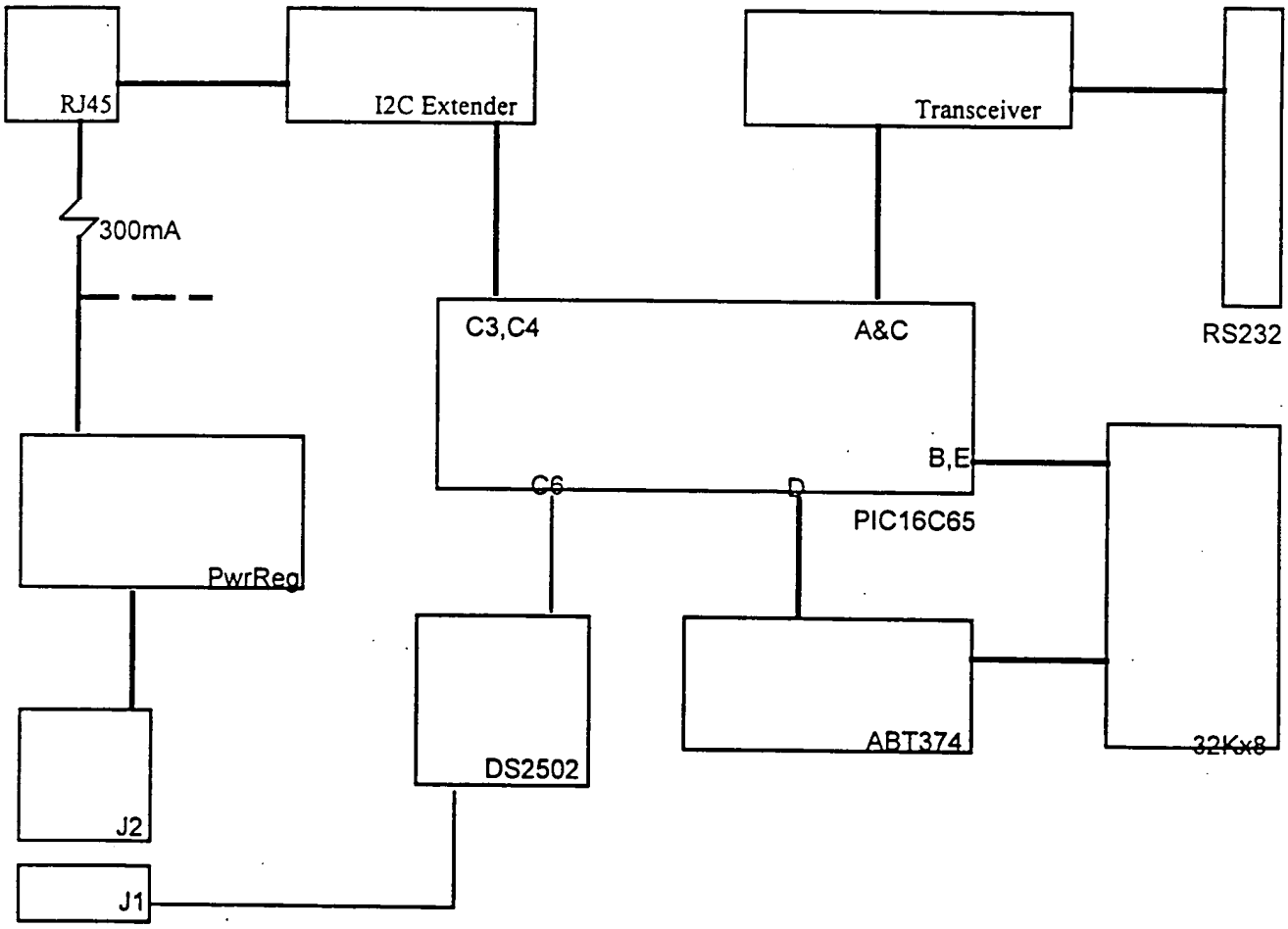


Figure 1: Remote Interface Interconnect

1.4 Serial ID EPROM

DS2502 is for storing board ID, connected to PIC16C65 Port C bit 6. The programming is handled through a jumper applied through connector J1. DS2502 is a signal powered, retaining the charge into a capacitor, sourced through the data line.

2.1 Alternative Power Source

The board is powered through 7.5V and 700mA (or 800mA which ever available) supply unit. After regulating the supply, it is an alternative source for the bias powered partition of the Raptor Wire Services. The bias powered block includes an NV-RAM and a PIC16C65 which are resident on the Raptor back plane.

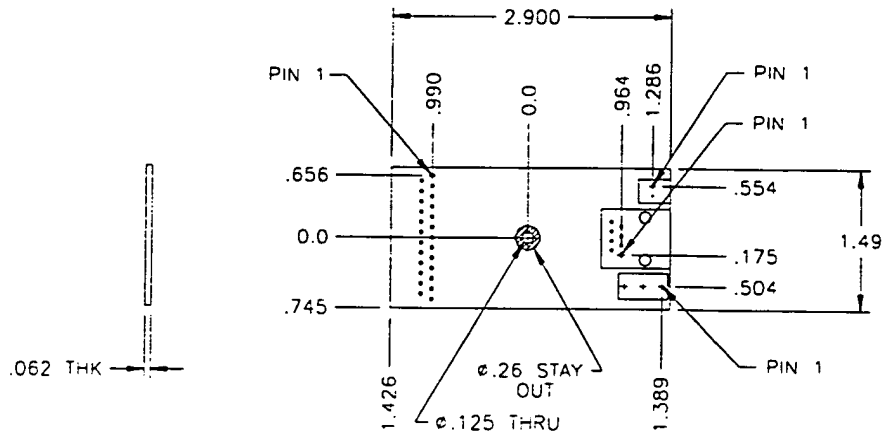
The power source is regulated through a high frequency switching regulator based on Linear Technology LT1376. The input to the regulator circuitry is off a wall mounted adapter. The regulated output is consumed locally and 300mA are sourced to the Raptor Wire Services through a fuse and an RJ45 P1.

2.2 Power Consumption

The following is an average estimated power consumption with the board running at a base frequency of 12MHz.

PIC16C65	Microcontroller	30mA
82B715	I2C Extender	10mA
32Kx8	SRAM	80mA
LT1133A	Transceiver	70mA
374	Latch	30mA
	Misc.	60mA
		280mA
	Alt. Source	300mA

	580mA
LT1376, L1, D2, D3, etc.	1.45 Watts max.



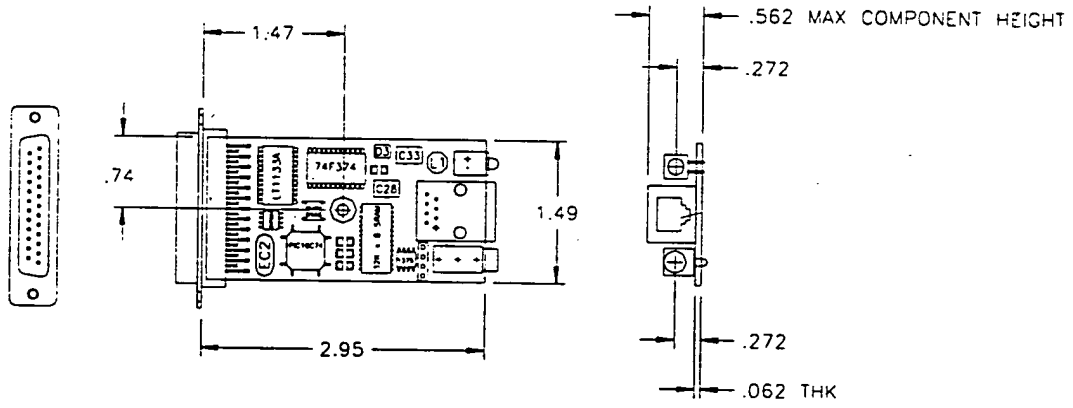


Figure 2: Mechanical Orientation

2.3 Board Layout

The board is based on controlled impedance of 60 Ohms +/- 10%, with 6 layers and test points for all signals. The width is restricted by the dimension of the RS232 due to the mounting constrains. The board is dual sided with active components kept on the top side only.

The high frequency bypass is kept with .1uf and .001uf, where the charge storage is kept by two 33uf and two 1uf capacitors.

The location and mounting of the power connector and the LED are kept such that the both sides of the cabinet are identical, therefore interchangeable.

3: Enclosure

The enclosure is planned to be Injection Molded Aluminum, a side view is in figure 3. Aluminum instead of plastic is selected due to the regulator heat, and EMI shielding.

The board connects at three locations between the top and the bottom enclosures. Two locations are based on the clamp shell design at the D-Sub and the RJ45, two opposite ends of the enclosure. The third location is a mounting hole in the center of the enclosure.

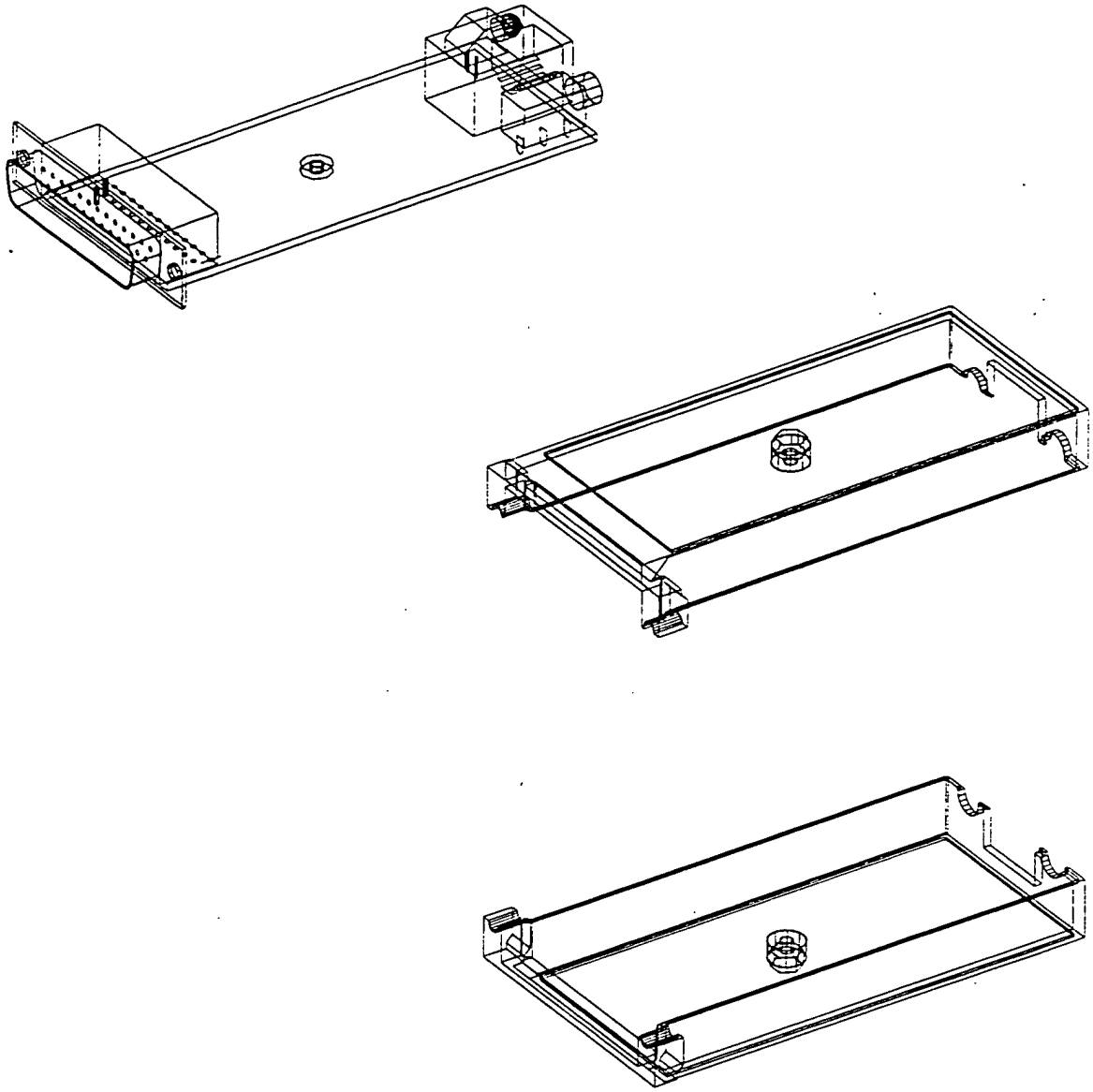


Figure 3: Board and Enclosure Isometric View

4.0 Environment

The environmental specification is based on assumptions:

The environment is Ground Fixed.
The "Quality Level of II" is used.
Bellcore Method I, Parts Count Method, Case 2 for prediction.
Burn-in time 120 hours.
Operated at 40C and 50% rated electrical stress.

4.1 Environmental Specification:

MEAN TIME BETWEEN FAILURE

2.445250e+06 Hours

This number is calculated based on the Bellcore Technical Reference TR-NWT-000332, Reliability Prediction Procedure for Electronic Equipment, Issue 4, September 1992.

ALTITUDE

Operating	-100 to 10,000 feet
Non-Operating	-100 to 40,000 feet

HUMIDITY

Operating	10% to 80% R.H., Maximum Gradient 10% per hour
Non-Operating	5% to 90% R.H., Maximum Gradient 10% per hour

TEMPERATURE (ambient)

Operating	10 to 40 degrees C Maximum Gradient 10 degrees C per hour
Non-Operating	-40 to 70 degrees C Maximum Gradient 10 degrees C per hour

SHOCK

Operating

Magnitude	2 G's (peak)
Duration	11 ms
Waveform	Half Sine

Non-Operating

Magnitude	10 G's (peak)
Duration	11 ms
Waveform	Half Sine

VIBRATION

Operating

Frequency Range	5 to 500 Hz
Magnitude	0.010 inch peak to peak displacement
Acceleration	0.20 GÕs peak

Non-Operating

Frequency Range	5 to 500 Hz
Magnitude	0.010 inch peak to peak displacement
Acceleration	0.50 GÕs peak

DROP (PACKAGED)
ASTM D4169

ELECTRICAL

Nominal Line	115 VAC or 230 VAC @ 50/60 Hz autoranging
Line Deviation	90-130 VAC & 180-256 VAC @ 47-63 Hz
Line Transient/Surge Susceptibility	1.25 x highest rated nominal voltage or 300 Vrms, whichever is less, for 1 second.

ELECTRO-MAGNETIC COMPATIBILITY

FCC, Class A under FCC Rule 15, Subpart B, conducted and radiated.

Canadian Radio Interference Regulations, C.R.C., c.1374, Sec. 2, as amended in The Canadian Gazette, Part II, Vol. 122, No. 20, dated Sept. 28, 1988.

European EMC Directive (89/336/EEC) CISPR 22 (Class B).

IEC 801-2:1984 8 kV air discharge

IEC 801-3:1984 3 V/m, 27-500 MHz

IEC 801-4:1988 1 kV mains, 500 V other.

SAFETY AGENCIES
UL, CSA, VDE, JIS

Electrostatic Discharge

Air Discharge	2.5 to 5.0KV	no errors allowed
	5.1 to 10.0 KV	recoverable errors through system allowed
	10.0 to 20.0KV	recoverable errors through power cycling allowed
Contact Discharge	0 to 8.0KV	recoverable errors through power cycling allowed

#11

Raptor Wire Service Architecture

Version 1.3

October 3, 1996

Prepared for
NetFrame Raptor Implementation Group

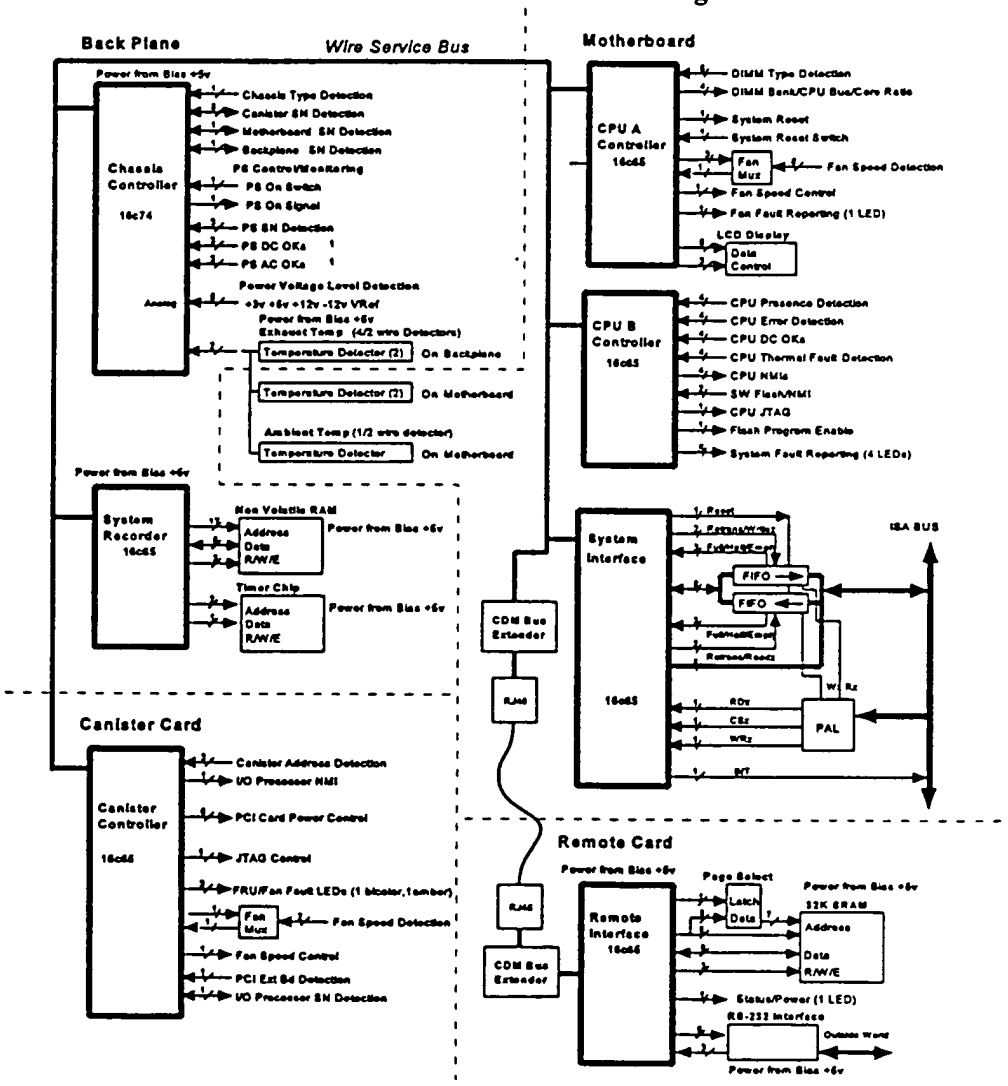
by
Karl Johnson (KJ)

EXHIBIT F

Introduction

"Wire Service" is the code name for the Raptor project system control, diagnostic and maintenance bus (formerly known as the CDM bus). Raptor is a completely "fly by wire" system - no switch, indicator or other control is directly connected to the function it monitors or controls. Instead, all the control and monitoring connections are made by the network of processors that comprise the "Wire Service" for the system. The processors are Microchip PIC processors and the network is a 400 kbps I^2C serial bus. A limited understanding of I^2C protocol is a prerequisite for understanding Wire Service protocols (See "The I^2C-bus and how to use it" - Philips Semiconductor, Jan 1992). Control on this bus is distributed, each processor can be either a master or a slave and can control resources on itself or any other processor on the bus.

Wire Service Hardware Block Diagram



Event Type

The event data type is to be used for alerting external interfaces of events in the Wire Service network. Event memory is organized as a bit vector. Each bit in the vector represents a particular type of event and there are a minimum of 16 bits in the vector.. Writing an event sets the bit representing the event in the bit vector. Whenever any bit in the event bit vector is non-zero, the interface indicates that events are present..

Reading the event type returns one or more event ID's, depending on request length and events actually pending in the interface. Once an event ID is read, the corresponding event bit is automatically cleared internally.

Request

Slave Address	Type/RW	N/A	N/A	Request 1-255

Response

Slave Address	Length N(1-32)	Event ID	. . .	Event ID	Status 0/Success

Write Event Message

Request

Slave Address	Type/RW	N/A	N/A	Length 1	Event ID (1-15)

Response

Slave Address	Length 0	Status 0/Success

Possible Event Types:
CPU Status Change
Power Status Change
Canister Status Change
Fan Status Change
Communications Queue Event

Wire Service Remote Interface Serial Protocol

The Wire Service Remote Serial protocol is used to communicate Wire Service messages across a serial link from a Wire Service Remote Interface processor attached to a Raptor to a Wire Service Remote management processor. It encapsulates Wire Services messages in a transmission envelope to provide error free communications and link security.

In order to be easily processed by the remote interface processor, the remote interface protocol is based on the concept of byte stuffing, That is certain byte values in the data stream always have a particular meaning and if that value must be transmitted as by the underlying application as data, it must be transmitted as a two byte sequence.

The bytes that have special meaning in the protocol are:
SOM - Start of a message
EOM - End of a message
SUB - The following byte in the data stream must be substituted before processing
INT - Event Interrupt

If any of these byte values occur as data in a message, a two byte sequence must be substituted for the byte. That sequence is a byte with the value of SUB followed by a byte with the value of the original byte incremented by one. For example if a SUB byte were to occcur in a message it would appear on the serial data stream as a SUB followed by a byte whose value is SUB+1.

Just as with Wire Service messages there are two classes of messages: 1) Requests - sent by remote managment systems (PC's) to the Wire Service remote interface and 2) Responses - returned to the requester by the Wire Service interface. The formats for the messages are as follows:

Request

SOM	Seq. #	TYPE	Data	...	Check	EOM

Response

SOM	Seq. #	STATUS	Data	...	Check	EOM

Event Interrupt

INT

Where:
SOM/EOM - Special data byte values marking the start and end of messages.
Seq # - A one byte sequence number that increments on each request and is copied in the response.

TYPE - One of the following types of requests:

- IDENTIFY- Request the remote interface to send back identification information about the system to which it is connected. This also resets the next expected sequence number. This message type does not require security authorization to be established first.
- SECURE - Establish secure authorization on the serial link by checking password security information provided in the message with the Wire Service system password.
- UNSECURE - Clear security authorization on the link and attempt to disconnect it. (Requires security authorization to have been established)
- MESSAGE - Take the data portion of the message and pass it to Wire Service for execution. The response from Wire Service is sent back in the data portion of the response. (Requires security authorization to have been established)
- POLL - Query the status of the remote interface. This is generally used to determine if an event is pending in the remote interface.

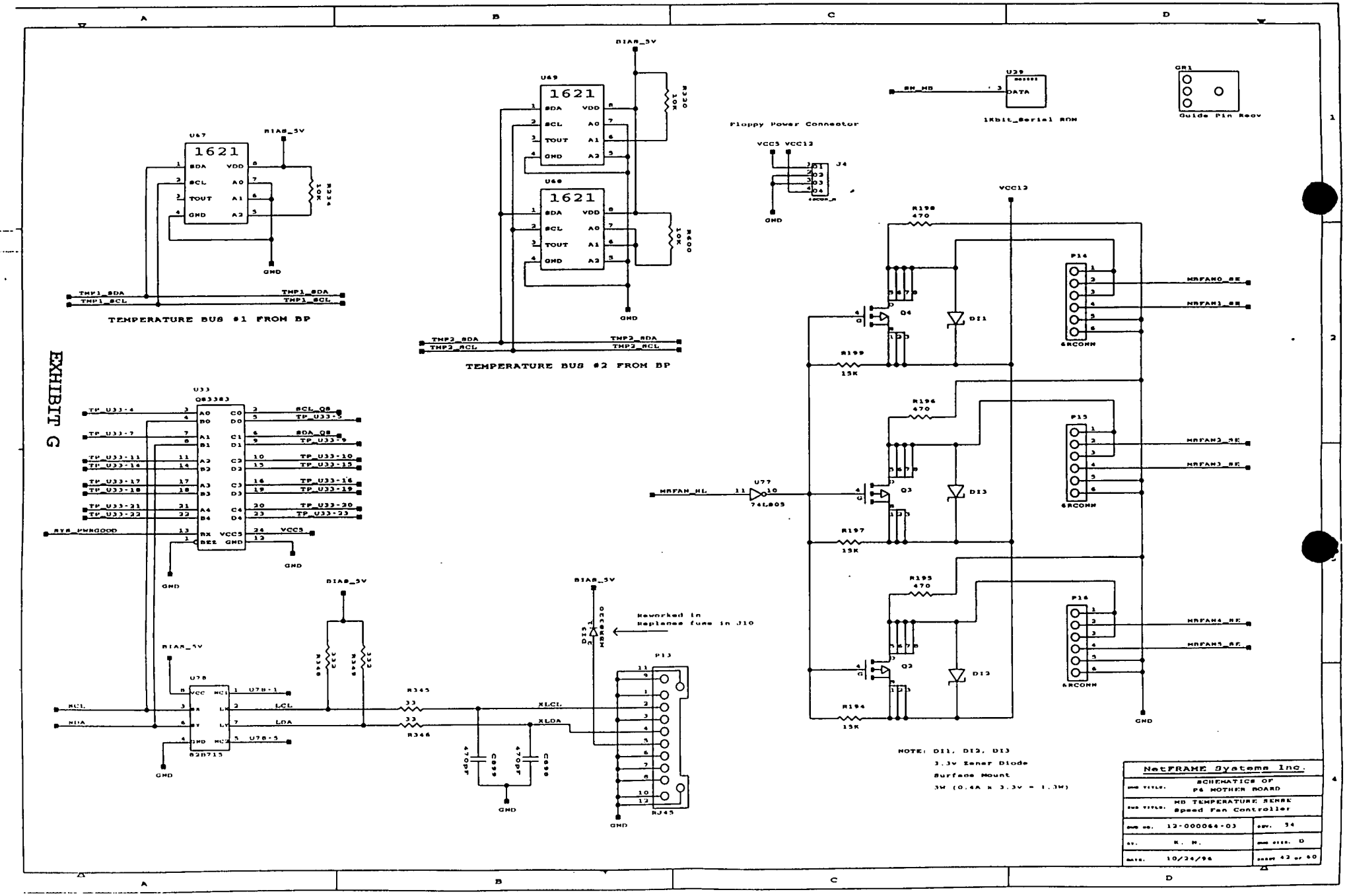
STATUS - One of the following response status values:

- OK - Everything relating to communication with the remote interface was OK
- OK_EVENT - Everything relating to communication with the remote interface was OK and there is one or more events pending in the remote interface.
- SEQUENCE - The sequence number of the request was not the current sequence number (retransmission request) or the next expected sequence number (new request). Sequence numbers may be reset by a IDENTIFY message.
- CHECK - The check byte in the request message was received incorrectly.
- FORMAT - Something about the format of the message was incorect. Most likley, the type field has an invalid value.
- SECURE - The message requires that security authorization be in effect, or if the message was of type SECURE, the security check failed.

Check - A message integrety check byte. Currently the value is 256 - the sum modulo 256 of all previous bytes in the message. (i.e. Adding all bytes in the message up to and including the check byte should produce a result of zero (0))

INT - A special one byte message sent by the remote interface when it detects the transistion from no events pending to one or more events pending. This message can be used to trigger reading events from the remote interface. Events should be read until the return status changes from OK_EVENT to just OK.

EXHIBIT G


TEMPERATURE BUS #1 FROM BP
TEMPERATURE BUS #2 FROM BP
Floppy Power Connector
1Kbit_Serial ROM
Guide Pin Recv
Reworked in Replace fuse in J10
NOTE: D11, D12, D13
3.3v Zener Diode
Surface Mount
3W (0.4A x 3.3V = 1.3W)
NetFRAME Systems Inc.
SCHEMATICS OF P6 MOTHER BOARD
MB TEMPERATURE SENSE 8Speed Fan Controller
12-000064-03
10/24/96

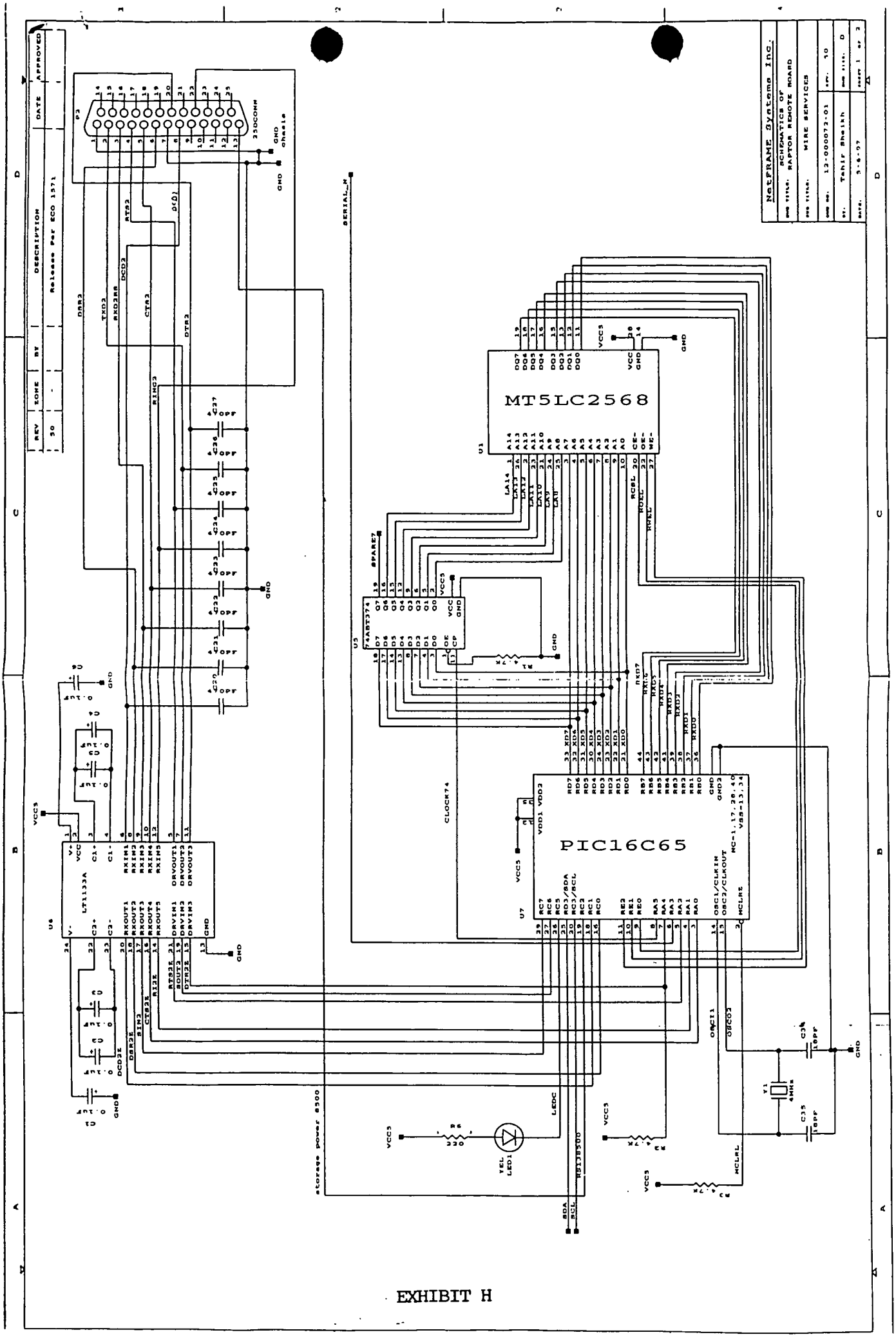
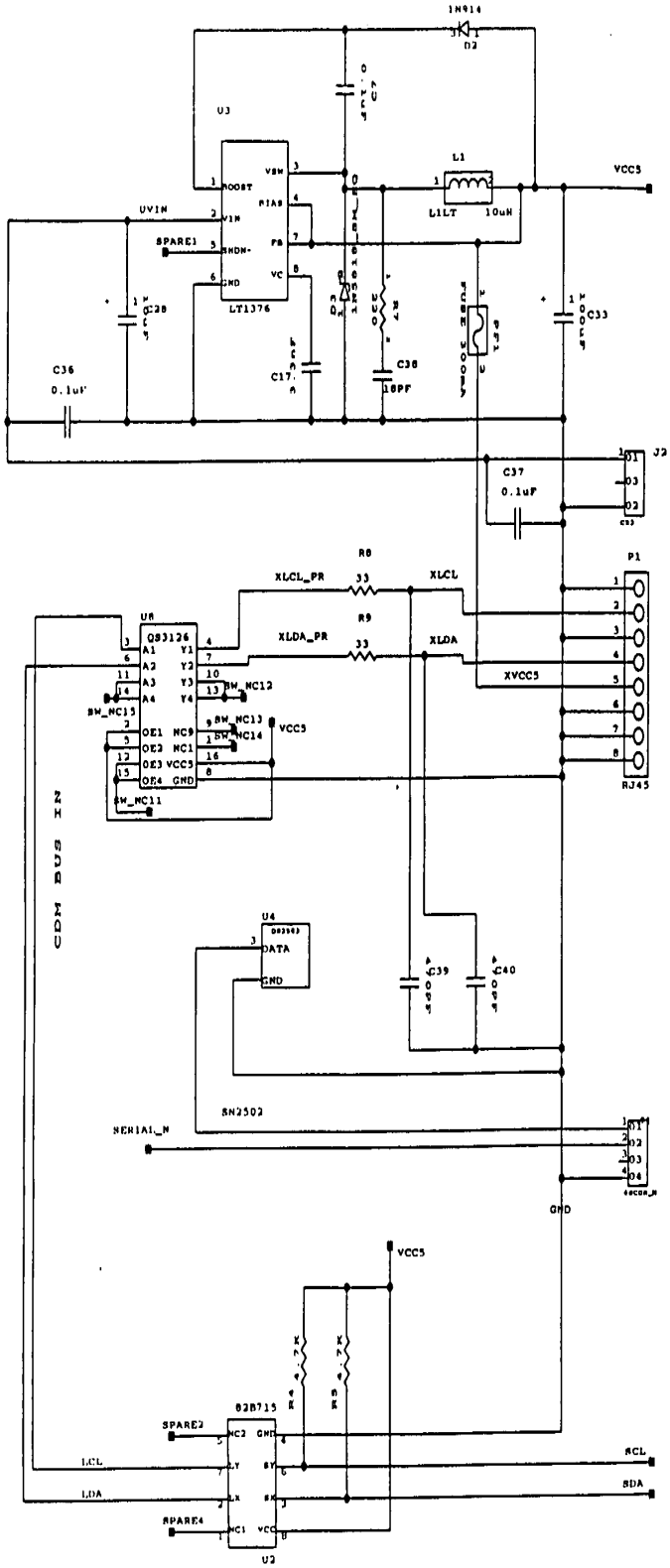
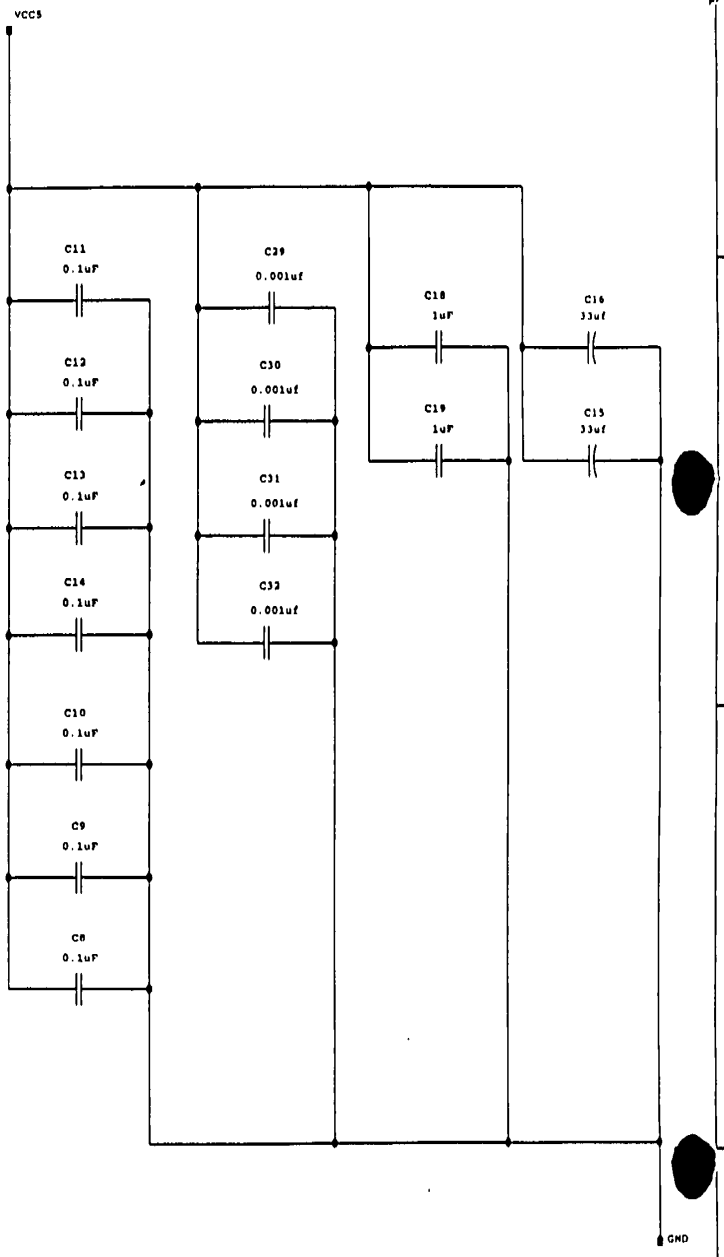

NetFRAME Systems Inc.
SCHEMATICS OF
RAPTOR REMOTE BOARD
WIRE SERVICES
12-000072-01
Tahir Sheikh
5-8-97
REV
ECME
BY
DESCRIPTION
DATE
APPROVED
Release Per ECO 1371
MT5LC2568
PIC16C65
LT1133A
74ABT374
SPARE7
SERIAL_M
CLOCK74
LEDC
storage power 8500
MCLRL
YEL
LED1
Y1
4MHz
GND
VCC5

EXHIBIT H





NetFRAME Systems Inc.	
SCHEMATICS OF RAPTOR REMOTE BOARD	
SUB TITLE: WIRE SERVICES	
DWG NO. 13-000072-01	REV. 50
BY: Tahir Sheikh	PWB SIZE: D
DATE: 5-6-97	SHEET 2 OF 2

Problem Statement

♦ Introduction

Maestro Recovery Manager(MRM) is a software which locally or remotely manage a Raptor when a server is down or up, operating system died, LAN communication failed, or other server components failed .

User will be able to manage the server in very simple, usable, and friendly GUI environment. MRM use modem for remote and serial communication port for local to communicate with server for diagnostic and recovery.

Primary role of remote management is diagnosing and restoring service as quickly as possible in case of a service failure.

System administrator, LAN administrator in customer shop and NetFrame Technical support will be primary user for the system.

♦ Requirement Sources

MRM requirements comes from the following

1 - Focus Group (Customer Support and Training.)
2 - User Walkthrough held by MRM team and Customer Support in Dec 96
3 - Down System Management Road map (96)
This road map is preliminary road map combined with Up System Management road map.
4 - MRM Road Map 97-98
This Road Map presented to Engineering Council Meeting on Mar 10, 1997.

5 - Raptor System, A Bird's Eye View.
6 - Raptor Wire Service Architecture

The following requirements have been identified for MRM

♦ Support Remote Management for Diagnostic and Recovery

Remote Management cover remote access to the Raptor Out Of Band management features. Remote Management will use Out of Band ,**Control Diagnostic and Monitor Subsystem (CDM)** remote management to cover the other high value added remote management functions. primary role of remote management is diagnosing and restoring service as quickly as possible in case of service failure.

EXHIBIT I

♦ Support Remote Management ... (continue)

The control of Raptor is completely "Fly By Wire" - i.e. no physical switch directly controls any function and no indicator is directly controlled by system hardware. All such functions referred to as "Out of Band " functions are controlled through a CDM. CDM basic functions are available so long as A/C power is available at the input to any of the power supplies.

CDM Subsystem supervises or monitors the following system features.

- **Power supplies** - Presence, status, A/C good, Power on/off and output voltage.
- **Environment** - Ambient and exhaust temperatures, Fan speed, speed control, Fan fault and overtemp indicators.
- **Processor** - CPU Presence, Power OK, Overtemp and Fault, NMI control, System reset, Memory type/ location and Bus/Core speed ratio.
- **I/O** - I/O canister insertion/removal and status indicator , PCI card presence, PCI card power and smart I/O processor Out Of Band control.
- **Historical** - Log of all events, Character mode screen image, and Serial number

♦ Support for Object Oriented Graphic User Interface

OO-GUI is graphic user interface with the following characteristic.

- **User task oriented**
  It uses tasks which user familiar and daily working with. User does not need to learn the tasks.
- **User objects**
  It uses objects which user working with during her or his daily work.
- **Simplicity and useablity**
  It is very simple to use and does not need long learning period.
- **Point and click with context sensitive help**
  Context sensitive help and point and click will help user to be very productive and get any information he needs on specific object or field or subject.
- **Drag and drop**
  Drag and Drop capability works with user object very well to accomplish the tasks.

♦ Release Requirements (MRM V2.0, 4Q96)

Maestro Recovery Manager (MRM)will support the following features locally through serial port and Wire Service Remote Interface card on the Raptor16.

MRM provide user friendly GUI with point and click capability to perform the following tasks which reviewed and accepted by the Focus Group for 4Q96 release.

- **Power On /Off**
  MRM support Power On/Off the server.
  User can do this task by right mouse click on the server object in the screen and see the result.

- **Display Flight Recorder.**
  While the server is working , Wire Service record all the server information in the 64K NVRAM. After the server failed, MRM will display the system log recorded in the NVRAM. User can evaluate the information and find the cause for the server failure. This can be done by right mouse click on the Flight Recorder object in the screen.

- **System Reset**

  MRM support rebooting the server by right mouse click on the server object in the screen. This is warm reboot of the server and works as pushing the "reset" button on the server.

- **Save**

  MRM will support saving Flight Recorder data, so user can send the file to the technical support for further diagnostic and recovery. It also can save the response for any Wire Service command failure.

- **On Line help**
  MRM will support online help contains overview, Getting Started, MRM tasks, Diagnostic and Recovery, and BIOS help.

- **B0 back plane support**

  MRM will support the server with B0 back plane . Server with B0 back plane display wrong time stamp. MRM uses NetWare 4.11 Operating system time stamp to display correct time stamp.